Министерство образования Республики Беларусь
Учреждение Образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных вычислительных средств

Лабораторная работа № 8
«ИЗУЧЕНИЕ ВОЛНОВОГО АЛГОРИТМА ТРАССИРОВКИ ПЕЧАТНЫХ СОЕДИНЕНИЙ»

Выполнили:
ст. гр. 850702
Маковский Р. А.
Турко В. Д.

Проверил:
Станкевич А. В.

Минск 2020

1 ИСХОДНЫЕ ДАННЫЕ

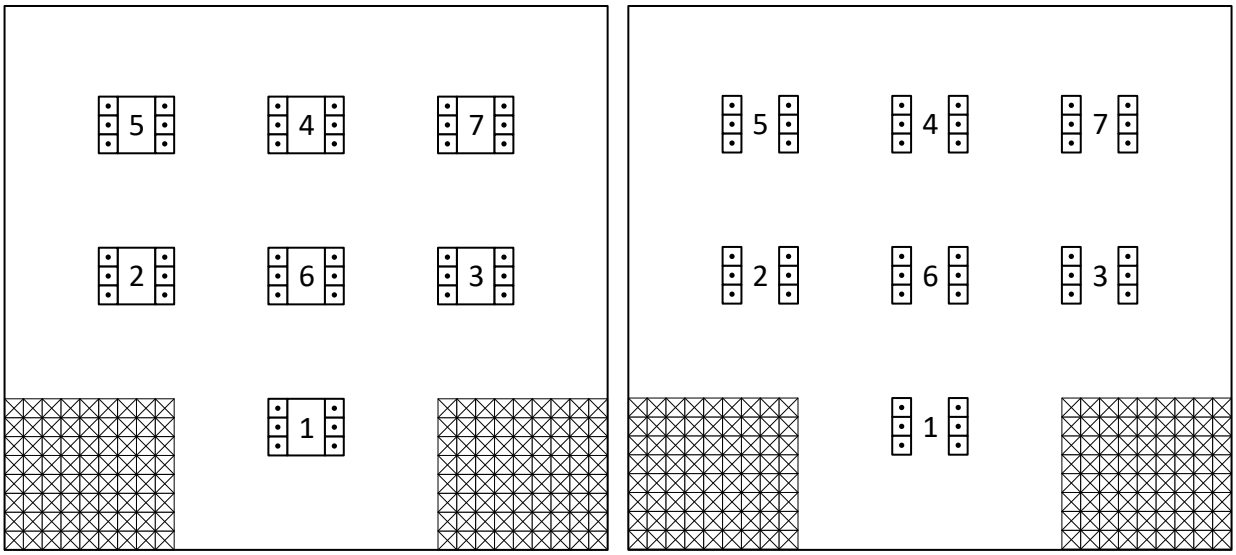
1.1 Условие задачи

Решить задачу трассировки печатных проводников с помощью волнового алгоритма для размещения элементов, полученного в предыдущей лабораторной работе. Трассировку выполнить в двух слоях.

1.2 Исходные данные

На рисунке 1.1. представлено размещение элементов на плате, а также ячейки, занятые корпусами и выводами элементов и запрещённые для прокладывания проводников.

На рисунке 1.2 представлена схема соединений элементов.



а) лицевая сторона б) обратная сторона

Рисунок 1.1 – Схема размещения элементов на плате.

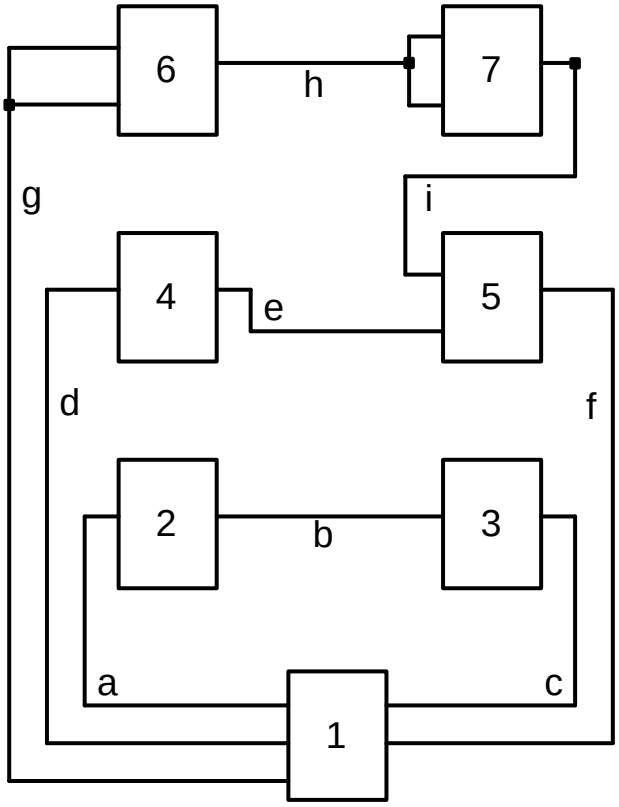


Рисунок 1.2 – Схема соединений элементов

2 ХОД РАБОТЫ

2.1 Представление данных

Каждый элемент (рис. 2.1а) содержит в себе координаты всех свои выводов. Каждая ячейка сетки (рис. 2.1б) содержит в себе свою координату и вес. Значение -1 говорит о том, что ячейка свободна, null – занята. Каждая трасса пути содержит в себе координаты своего начала и конца.

```
class Element {
  Coord first;
  Coord second;
  Coord third;
  Coord fourth;
  Coord fifth;
  Coord sixth;

  Element(
      {this.first,
      this.second,
      this.third,
      this.fourth,
      this.fifth,
      this.sixth});
}
```

а) элемент схемы

```
class Cell {
  int weight;
  Coord coord;

  Cell.empty(this.coord) : weight = -1;
  Cell(this.coord, this.weight);
  Cell.occupied(this.coord) : weight = null;
}
```

б) ячейка сетки

```
class Line {
  Coord start;
  Coord end;

  Line(this.start, this.end);

  int get length => this.start.distanceTo(this.end);
}
```

в) путь соединения элементов

Рисунок 2.1 – Представление данных на ЯП

2.2 Основные использованные функции

На рисунке 2.2 представлена функция, распространяющая очередной фронт волны. В данной лабораторной работе использовался следующий приоритетный порядок проведения пути: слева, снизу, справа, сверху.

На рисунке 2.3 представлена функция, проверяющая выбранную ячейку. Если её координаты совпадают с координатами конца трассы (приемника волны), то возвращается null. В противном случае если ячейка занята, возвращается false, а если не занята, то ячейке присваивается вес и возвращается true.

```
bool _spreadWave(Matrix<Cell> matrix, Coord currentCoord, Coord end) {
  bool resultD;
  bool resultL;
  bool resultR;
  bool resultU;
  bool foundEnd = false;
  int i = 0;
  while (!foundEnd) {
    int found = 0;
    matrix.forEach((item) {
      if (item.weight == i) {
        Coord currentCoord = item.coord;
        resultR = _checkCell(matrix, currentCoord.right(), end, i + 1);
        resultR == null ? foundEnd = true : found++;
        resultU = _checkCell(matrix, currentCoord.up(), end, i + 1);
        resultU == null ? foundEnd = true : found++;
        resultL = _checkCell(matrix, currentCoord.left(), end, i + 1);
        resultL == null ? foundEnd = true : found++;
        resultD = _checkCell(matrix, currentCoord.down(), end, i + 1);
        resultD == null ? foundEnd = true : found++;
      }
    });
    if (found == 0) { return false; }
    i++;
  }
  return foundEnd;
}
```

Рисунок 2.2 – Функция, реализующая распространение очередного фронта волны

```
bool _checkCell(Matrix<Cell> matrix, Coord currentCoord, Coord end, int i) {
  if (matrix.isValidCoord(currentCoord)) {
    if (matrix[currentCoord.y][currentCoord.x].weight == -1) {
      matrix[currentCoord.y][currentCoord.x].weight = i;
      if (currentCoord.x == end.x && currentCoord.y == end.y) {
        return null;
      }
      return true;
    } else { return false; }
  } else { return false; }
}
```

Рисунок 2.3 – Функция проверки ячейки

```
int _buildConnection(Matrix<Cell> matrix, Coord start, Coord end, String name)
{
  bool resultD;
  bool resultL;
  bool resultR;
  bool resultU;
  Coord currentCoord = end;
  int length = 0;
  while (currentCoord.x != start.x || currentCoord.y != start.y) {
    resultD =
_buildConnectionSegment(matrix, currentCoord, currentCoord.right(), name);
    if (!resultD) {
      resultL =
_buildConnectionSegment(matrix, currentCoord, currentCoord.up(), name);
      if (!resultL) {
        resultR =
_buildConnectionSegment(matrix, currentCoord, currentCoord.left(), name);
        if (!resultR) {
          resultU =
_buildConnectionSegment(matrix, currentCoord, currentCoord.down(), name);
          if (resultU) {currentCoord = currentCoord.down(); length++;}
        } else {currentCoord = currentCoord.left(); length++;
      } else {currentCoord = currentCoord.up(); length++;}
    } else {currentCoord = currentCoord.right(); length++;}
  }
  matrix[start.y][start.x].occupy();
  matrix[start.y][start.x].key = name;
  return length;
}
```

Рисунок 2.4 – Функция, реализующая построение пути

```
bool _buildConnectionSegment
(Matrix<Cell> matrix, Coord cur, Coord to, String name) {
  bool success = false;
  if (matrix.isValidCoord(to)) {
    if (matrix[to.y][to.x].isFree) {
      int curWeight = matrix[cur.y][cur.x].weight;
      int toWeight = matrix[to.y][to.x].weight;
      if (toWeight < curWeight) {
        matrix[cur.y][cur.x].occupy();
        matrix[cur.y][cur.x].key = name;
        success = true;
      }}}
  return success;
}
```

Рисунок 2.5 – Функция, реализующая построение части (одной ячейки) пути

2.3 Результат трассировки

При построении проводника учитывалось, что проводник должен иметь минимальную длину и не иметь пересечений с другими проводниками в своём слое. Приоритет проведения проводника учитывался последовательностью просмотра ячеек, соседних с текущей.

Для построения очередного проводника распространение волны проводилось по двум сторонам платы, после чего из двух полученных проводников выбирался кратчайший.

На рисунке 2.6 представлен результат трассировки. Чёрной сплошной линией обозначены проводники, проведенные по лицевой стороне платы, зелёной пунктирной – по обратной.

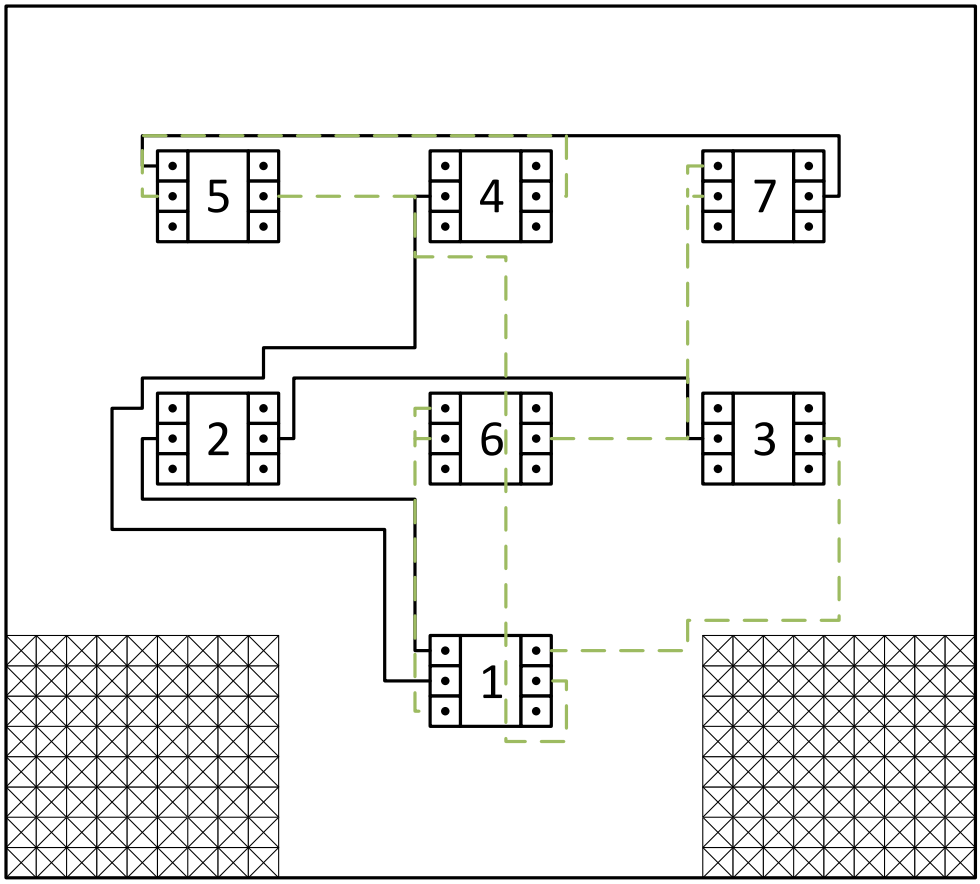


Рисунок 2.6 – Результат трассировки

3 ВЫВОД

В лабораторной работе мы познакомились с волновым алгоритмом Ли для трассировки печатных соединений, а также с некоторыми его модификациями. Провели трассировку печатных соединений в прямоугольной системе координат в двух слоях печатной платы с помощью алгоритма Ли. Для схемы с небольшим количеством элементов и соединений можно использовать волновой алгоритм Ли без модификаций, однако при большом количестве элементов и соединений имеет смысл использовать модификации данного алгоритма, которые позволяют уменьшить объём требуемой памяти